A Method and System for Discovery of Trades Between Parties

December 6, 2000

1 Related Applications

This application claims priority to provisional application no. 60/168,754 filed on December 6, 1999, titled, "An E-Commerce Infrastructure for Value Chains", the contents of which are herein incorporated by reference.

2 Field of the Invention

The invention relates to a method and system for discovery of trades between parties. In particular, the invention is a system which allows buyers to define their preferences and sellers to define their capabilities, then determines which trading points maximize the utility of the buyer. The system suggests trades by exploiting the flexibilities and tradeoffs encoded by both parties, thus providing win-win trades. A second level of optimization ranks the trades with all suppliers, allowing the buyer to rapidly determine the best alternatives. The system allows for rich negotiation spaces and supports continuous, discrete, and range or interval decision factors.

3 Background of the Invention

The present invention relates to methods of automatic exploration and exploitation of the flexibilities possessed by negotiating parties to uncover improved win-win agreements. The invention describes computationally efficient mechanisms that are applicable whether there are one or many selling parties. The precise number and types of negotiating dimensions are irrelevant as long as they are numerical. Thus the present invention applies equally to the optimal determination of terms in the purchase of a commodity or an arbitrarily complex artifact.

Electronic markets have proliferated over the last few years with the advent of B2C (business-to-consumer) and B2B (business-to-business) electronic commerce. Such market places have yielded significant cost savings

by lowering the transaction costs between buyers and sellers. Buyers have also profited through increased competition between suppliers. However, electronic markets have hurt suppliers, since the zero-sum negotiation over price has been at their expense. The present invention describes a tool whereby cost savings for both parties are derived from the discovery of win-win trades. Fundamentally, the system works by allowing trading parties to describe their desired trade across multiple dimensions and to express their flexibility around this ideal trade. Through an algorithmic exploration of their flexibilities, the present invention can discover trades that are near the ideal trades of both parties, enabling both to win.

The adoption of B2B and B2C electronic commerce was facilitated by the migration of catalogues online. This familiar method of presentation ameliorated the significant cultural change to electronic trade. For the foreseeable future, electronic commerce will be dominated by online catalogs. At present, online catalogues are direct translations of their hardcopy counterparts where the attributes of a product are described and a price quoted. Inevitably however, online catalogs will become more expressive. Catalog entries will be able to represent price breaks for large quantity orders, lot sizes, etc. Thus it is important that any software (like the present invention) that uncovers mutually beneficial trading scenarios is able to operate with such catalogs. Consequently, in the present invention there is an asymmetry between buyer (usually a human) and seller (usually an online catalog).

One of the reasons catalogs have come to dominate electronic commerce is that the types of goods that can be represented in catalogs are simple. Whether the product is pens or paper clips, different vendor's offers differ little from each other (a pen is a pen is a pen), and a quick scan of a catalog gives a buyer enough information to make an informed purchase. These types of goods are low margin and inexpensive. In contrast, the vast amount of purchasing between businesses involves materials which are directly connected with business operations - car parts, turbines, etc. Such direct goods are the future of electronic commerce. Unlike present-day engines, any truly useful procurement tool must be able to support direct materials with complex attributes and complex inter-relationships between its components.

Electronic commerce offers unprecedented opportunities for more informed decision-making for both buyers and sellers. The past few decades have seen the widespread adoption of enterprise resource planning (ERP) systems, to the point that now almost every major company has some form of ERP software. ERP functions as the digital nervous system of a company, transmitting and logging information between the company's many different business functions. ERP software keeps track of inventory, monitors the state of purchase orders, signals when a company should reorder direct and indirect materials, and a myriad of other functions. Consequently, ERP databases are a rich source of information to optimize a company's operations. Yet today this information is rarely used to make more informed buying and selling decisions. The present invention can utilize such information sources to optimize a company's interactions with suppliers and customers.

One important manner in which this optimization can occur is through an analysis of *all* cost factors. Current buying and selling practices often focus on limited goals, e.g., minimize the total purchase price. Myopic purchasing strategies often result in higher total cost of ownership when all cost factors relevant to a product in its lifetime of use are included. These other cost factors can be significant. Why save the money in taking delivery two days late if the receiving docks will be full at that time and an additional shift needs to be hired to clear the docks? Why order the cheaper drill bit if it is much more expensive to replace when it breaks? The present invention improves trades by minimizing the total cost of ownership of a product, yielding significant savings to its users. Many total cost factors are difficult to quantify – e.g. what is the cost of dealing with a unionized versus a non-unionized supplier? Consequently, the present invention supports qualitative (best guesses and intuition) as well quantitative factors.

All companies are situated in a supply or value chain. At each step in the chain, a company purchases from its suppliers, transforms these inputs, and sells the output to its customers. The termination of the supply chain is the sale of the final product to the end consumer. Since the only influx of external capital comes from the end consumer, companies have realized that they compete not only as individuals but also as entire supply chains. As result, software products have recently become available which attempt to streamline the operations of links within the entire supply chain. This software, variously called supply chain optimization (SCO) or advanced planning and optimization (APO), operates on the basis of forecasted demand at various points within the supply chain. Based on these predictions, plans are generated telling companies how much to produce and how to schedule their operations. SCO systems are a valuable source of intra-company information – data the present invention capitalizes on. Because SCO software relies on forecasted demand, it is only as helpful as the forecast is accurate, and, unfortunately, in many cases demand is very difficult to predict. How can the software know that laundry detergent will go on special at grocery stores in the Northeast in 7 weeks? As a result of the volatility in demand and the many other unpredictable perturbations that plague supply chains, companies keep significant buffers in the form of inventories. In addition to planning, businesses must also be able to adapt to unplanned effects. Such adaptation requires flexibility and a means to exploit that flexibility. The present invention exploits the flexibility of trading parties to streamline the operations of supply chains by smoothing the boundaries between trading parties.

The present invention is therefore a system to allow trading parties to express trading desires and constraints across many and varied different factors. These trading preferences are informed by many different data sources to optimize for a company's internal operations and its connections to it's supply chain through an analysis including total cost factors. The flexibility expressed by all trading parties is exploited to locate win-win opportunities for all parties if they exist.

4 Summary of the Invention

We describe the present invention in its application to facilitating trade between buyers and sellers, but note that the mechanisms described are much more general. We can easily imagine, for example, using the present invention to match individuals (with the desires and skills) to projects.

The inspiration for the present invention comes from utility theory developed by economists since the 1960's. Since we are interested in multiple dimensions of negotiation, we draw from the multi-attribute utility theory literature.[1] Utility is an abstract concept which has been formalized in various ways. For the present purposes utility, u, is a number between 0 and 1 representing a party's willingness to trade. Larger values indicate a greater willingness.

4.1 The Negotiation Space

In any negotiation the parties must come to agreement on the factors requiring negotiation. We call these factors dimensions or variables. As an example, when purchasing a car, the buyer may be concerned with price, time of delivery, and color. Each factor price, time, and color is a dimension. Most dimensions can be classed as one of three types: continuous, discrete, or range/interval. A continuous dimension is one like price for which the buyer's utility varies smoothly across that dimension. The buyer's utility at \$23 001.00 is almost the same as the utility at \$23 000. Color is a discrete dimension. Since the car may only be available in black, white, and silver, the domain of this dimension is the finite set of values {black, white, silver}. Moreover, the buyer's utility may be quite different for the three colors. The third class of dimensions is called interval dimensions. An interval dimension arises often in B2B negotiations. If a machined part is built to some tolerance (e.g., the inner diameter of a screw is between 24.5 and 25.5 mm), the range of variability in the dimension is specified as an interval. In the language of statistical quality control, a certain percentage of the machined parts will fall in this range. These three broad classes of variables capture almost all the types of attributes relevant to B2B negotiation.

The present invention operates over any number of continuous, discrete, and range or interval variables. We call the negotiation space X and any point in the negotiation space $(\mathbf{x}, \varkappa, \mathbf{r}) \in X$. It is important to recognize that the single trading point $(\mathbf{x}, \boldsymbol{\varkappa}, \mathbf{r})$ may have multiple components, e.g., price = \$23 000, time of delivery = 3 weeks, color = black.

In the present invention, the space of negotiation is agreed upon by all parties involved prior to the commencement of any negotiation. We can, however, imagine more dynamic situations in which dimensions are introduced and discarded over time.

[1]For a good introduction to multi-attribute utility theory see [1].

4.2 The Buyer's Utility Function

A party defines it's utility function over this space so that every $(\mathbf{x}, \boldsymbol{\varkappa}, \mathbf{r})$ is assigned a utility number indicating the party's willingness to trade. We indicate the utility function as $u((\mathbf{x}, \boldsymbol{\varkappa}, \mathbf{r}))$. A great deal of work has been done on the appropriate form for utility functions. In the present invention, we take a simple form for the utility function for two reasons. First, we would like the form of the utility to be conducive to rapid computation. Second we would like the utility to be simple enough to be easily understood by and elicited from users of the invention. With no loss in generality, we write the utility function as $u((\mathbf{x}, \boldsymbol{\varkappa}, \mathbf{r})) = \exp(-d((\mathbf{x}, \boldsymbol{\varkappa}, \mathbf{r})))$ where $d(\mathbf{x})$ is interpreted as the distance of trading point $(\mathbf{x}, \boldsymbol{\varkappa}, \mathbf{r})$ from the most preferred trade.

So that we can operate against seller catalogs, only the buyer needs to define a utility function. Across the continuous dimensions, the buyer's utility is defined by specifying the most preferred (or ideal) continuous dimensions and the manner in which utility drops off as we move away from this ideal. For the discrete dimensions, the utility is specified in tabular form since there are a finite number of alternatives. Again, the buyer must specify it's ideal discrete values and how utility decays away from those values. In section 6.1 we describe how this is accomplished. The range dimensions contribute to utility similarly; the buyer specifies an ideal range and the utility decays for ranges other than the ideal according to their distance from the ideal.

The utility function can also express tradeoffs between variables, e.g., I may take delivery in 5 weeks if the price drops to \$20 000, or I may accept the white car if I can take delivery in 2 weeks. The tradeoffs may be between pairs of continuous dimensions (as in the first case), between pairs of discrete variables, or between continuous and discrete variables (as in the second case).

4.2.1 Normalization and Weighting

When utility is defined over different types of variables, it is important to normalize the contributions of each variable so that the buyer can weight the importance of the various contributions to utility. This is a difficult problem. How should a buyer's color preferences be normalized so that they can be traded off against time of delivery? The present invention solves this problem by requiring that the average distance of any negotiation variable from its ideal value is the same for all dimensions. Since the buyer is more interested in those regions of the negotiation space where the utility is high, the average is weighted by utility. This procedure defines a manner in which to define a baseline where all dimensions contribute equally. Given this baseline, the buyer can then weight the various contributions and obtain useful results.

4.2.2 Utility Elicitation

Since utility is fundamental to the present invention, its elicitation from the buyer is important. Utility may be defined using any of a number of sources:

1. graphical user interfaces associated with the invention
2. standard benchmark criteria applicable to the domain
3. formal methodologies like the analytical hierarchical process [2], or discrete choice analysis [3]
4. inferred through models

We expand briefly upon method 4. As discussed in the background section, it is important to buyers to minimize their total cost of ownership. If we have a function representing these costs as a function of the negotiation variables, and perhaps other factors, this function can be used to infer a utility function which will act to minimize the total costs. Later we describe how this can be accomplished.

4.3 A Supplier's Capabilities

As noted previously, suppliers are treated differently from buyers so that the tool can operate against catalog information with no human intervention required on the part of the seller. In fact, we do not require sellers to define a utility at all.

A supplier cannot offer deals at all points within the negotiation space X, e.g., he certainly can't offer the black car tomorrow for free! A capability then represents the ability of a supplier to deliver and defines a subspace of X. It can include such things as price discounts on large volume orders, variation in delivery time as a function of price, etc. Since these relationships are already specified by businesses in terms of simple rules like *"the price per unit is $10.00 if 1 to 999 units are ordered and $9.50 per unit if 1000 or more units are ordered"*, suppliers' capabilities are represented in the present invention by piecewise linear functions.

4.4 Negotiation Constraints

Both parties may have constraints which must be satisfied in order for them to trade. For example, the buyer may not buy the car unless he gets it within 6 weeks, or he may not purchase the car if it is available only in white. These are examples of continuous and discrete constraints, respectively. A continuous constraint sets a requirement on the continuous variables. In the present invention, continuous constraints must be either linear or quadratic. Discrete constraints involve discrete variables. A discrete constraint can be expressed as a list of the allowed (or disallowed) combinations of the discrete variables for which the trade will be acceptable. For example, if the buyer would accept either the black or the silver car, the constraint would

list both these colors as viable. It is important to note that both continuous and discrete constraints may involve one or more variables. We can also express constraints involving both types of variables by allowing the continuous constraints to differ depending on the discrete variables.

4.5 Utility Optimization

With the major components of the invention in place, we describe how the overall system works. As a procurement tool for the buyer, there are two levels of optimization. First, for any given supplier we maximize the buyer's utility, subject to the supplier's capabilities to find that trade which makes the buyer as happy as possible. Since we are optimizing within a supplier's capabilities, the supplier has expressed a willingness to complete the trade at whatever point is determined to be optimal. The tool then optimizes across suppliers to rank them according to utility at the optimal point. A graphical user interface allows a buyer to investigate the trades suggested by the tool by sorting according to any dimension or by the overall utility.

Utility, while a useful concept in assessing an overall score, may be of limited use to a buyer due to its abstract meaning. Consequently, we can also apply the total cost of ownership function to the results to rank order the suggested trades according to their various cost components. Recall that for any trade $\mathbf{x} \in X$, the total cost of ownership function returns the various cost contributions. This additional information aids the buyer in his purchasing decision. The utility number for each trade is still useful because the total cost of purchase function includes only those cost factors which can be quantified, whereas the utility also includes "softer" qualitative factors.

4.5.1 Aggregation

In addition to optimizing against one supplier at a time, the present invention can also be used to optimize against an arbitrary aggregation of suppliers. This is important if, for example, no single seller can supply the large volume requested by a buyer. In this mode of operation, the buyer specifies sets of suppliers participating in the aggregation and the dimensions over which aggregation can occur, and the tool finds the optimal combination in which to distribute the volume dimension over the allowed suppliers.

5 Brief Description of the Figures

Figure 1 shows an an architecture for the invention.

Figure 2 shows a schematic of a buyer-specific capability with examples indicating potential input.

Figure 3 shows a schematic of a supplier-specific preference with examples indicating potential input.

n_c	number of continuous dimensions
n_d	number of discrete dimensions
n_r	number of range dimensions
$\mathbf{x}$	n_c-vector of values for continuous dimensions
$\boldsymbol{\varkappa}$	n_d-vector of values for continuous dimensions
x_i	value of ith continuous variable
$\varkappa_i$	value of ith discrete variable

Table 1: Definition of the negotiation search space.

6 Detailed Description

6.1 Theory

In this section we outline the mathematical foundations of the optimization process in sufficient detail to allow for computer implementation.

6.1.1 The Negotiation Space

In Table 1 we define the parameters which collectively define the space of negotiation X. For each of the n_c continuous variables, we specify an allowed range over which that continuous dimension may vary as $x_i \in X_i = [\underline{x}_i, \overline{x}_i]$, where $\underline{\mathbf{x}}$ is the n_c-vector of lower continuous bounds and $\overline{\mathbf{x}}$ is the n_c-vector of upper continuous bounds. Each discrete variable assumes a value from within its domain $\varkappa_i \in \mathcal{D}_i$. Without loss of generality, we label the domain of discrete variable i by $\mathcal{D}_i = [1, \cdots, d_i]$ where $d_i \geq 0$ is an integer giving the number of possible values discrete variable $\varkappa_i$ may assume.

With these definitions, we define the space of negotiation by the tensor product $X = X_1 \otimes \cdots \otimes X_{n_c} \otimes \mathcal{D}_1 \otimes \cdots \otimes \mathcal{D}_{n_d}$. Range variables are treated separately and not negotiated over.

6.1.2 The Utility Function

The utility function is a mapping from X into the interval $[0, 1]$. As indicated earlier we assume the utility to have the form $u(\mathbf{x}, \boldsymbol{\varkappa}) = \exp[-d(\mathbf{x}, \boldsymbol{\varkappa})]$ where $d(\mathbf{x}, \boldsymbol{\varkappa})$ is interpreted as a distance. In what follows we will assume that in its simplest form the distance function has the form

$$d(\mathbf{x}, \boldsymbol{\varkappa}, \mathbf{r}) = \big(\mathbf{x} - \boldsymbol{\mu}(\boldsymbol{\varkappa})\big)^t \mathbf{C}^{-1}(\boldsymbol{\varkappa})\big(\mathbf{x} - \boldsymbol{\mu}(\boldsymbol{\varkappa})\big) + Z(\boldsymbol{\varkappa}) + R(\mathbf{r}; \boldsymbol{\varkappa}).$$

Each contribution to the distance function is positive. We consider each contribution to the distance in turn, beginning with the range variable contribution $R(\mathbf{r}; \boldsymbol{\varkappa})$.

First, we note that the range distance depends on the setting of the discrete variables. This allows the buyer to express different preferences for the range variables depending on discrete factors. The total range distance is summed up over all possible range variables so that $R(\mathbf{r};\varkappa) = \sum_{i=1}^{n_r} R_i(r_i;\varkappa)$. The vector $\mathbf{r}$ indicates the preferred values for all range variables. If range variable i is specified as the interval $r_i \equiv (\underline{r}_i, \overline{r}_i)$ (where $\overline{r}_i > \underline{r}_i$) then $\mathbf{r}$ is an n_r-vector of such tuples. The distance contribution, R_i, from one range variable will depend on the application. If the range variables are meant to represent the tolerances on machined parts where issues of statistical quality control are important, then the distance between two ranges might be related to the overlap between Gaussian distributions. If r_i is interpreted as a Gaussian having mean $(\underline{r}_i + \overline{r}_i)/2$ and standard deviation proportional to $\overline{r}_i - \underline{r}_i$ then an appropriate range distance is given in Appendix A. Other choices for the range distance function are certainly possible.

The continuous distance is quadratic and determined by the positive semidefinite $n_c \times n_c$ matrix $\mathbf{C}^{-1}$. We have allowed this matrix to vary with the setting of the discrete variables and indicated this explicitly through $\mathbf{C}^{-1}(\varkappa)$.[2] The n_c-vector $\boldsymbol{\mu}$ may also depend on $\varkappa$ and indicates the point at which the utility is maximal – $\boldsymbol{\mu}$ is thus identified with the ideal value for the continuous variables. The precise quadratic form is convenient, but, using recent developments in interior point methods, other convex functions are also computationally tractable [4].

The discrete distance is determined through the function $Z(\varkappa)$ which maps the discrete space $\mathcal{D}_1 \otimes \cdots \otimes \mathcal{D}_{n_d}$ onto the positive real line $[0, \infty]$. In keeping with the assumption that distance is a function of only pairs of components x_i, x_j, we assume the discrete distance has the form[3]

$$Z(\varkappa) = \sum_{i=1}^{n_d} \Big\{ Z_i(\varkappa_i) + \sum_{j=1(\neq i)}^{n_d} Z_{i,j}(\varkappa_i, \varkappa_j) \Big\}.$$

Each contribution $Z_{i,j}$ is a table consisting of $d_i d_j$ entries, where $Z_{i,j}(\varkappa_i, \varkappa_j)$ can be interpreted as the distance if discrete dimension i has value $\varkappa_i$ conditioned on discrete dimension j having value $\varkappa_j$. The diagonal terms Z_i offer an unconditional distance. The most preferred value for the ith discrete dimension is that for which $Z_i(\varkappa_i) = 0$.

Rather than require the user to enter the distances explicitly, there are numerous ways in which the distances can be generated automatically based upon a buyer's ranking of preferred values. Further details can be found in Appendix B.

Weighting of Dimensions

In many cases it is important for simple modifications of the distance function to re-weight the contributions to the total distance. If $\mathbf{w}_c$ is an n_c-vector of weights for the continuous dimensions, we can accomplish this by letting $\mathbf{C}_w^{-1} = \mathbf{W}_c \mathbf{C}^{-1} \mathbf{W}_c$ where $\mathbf{W}_c$ is the diagonal matrix $\mathbf{W}_c = \mathrm{diag}(\mathbf{w}_c)$.[4] In a similar way we modify the

[2] Where no confusion will arise we eliminate the dependence of $\mathbf{C}^{-1}$ on $\varkappa$ for notational simplicity.

[3] Later we shall generalize this distance to include weighting of dimensions.

[4] $\mathbf{M} = [m_{i,j}] = \mathrm{diag}(\mathbf{v})$ is the diagonal matrix formed by setting $m_{i,i} = v_i$ and $m_{i,j} = 0$ for $j \neq i$.

discrete distance to $Z_{w,i,j}(\varkappa_i, \varkappa_j) = w_{d;i}w_{d;j}Z_{i,j}(\varkappa_i, \varkappa_j)$ where $\mathbf{w}_d$ is the n_d-vector of weights for the discrete variables and $w_{d;i}$ is its ith component. The range contribution is also modified so that $R_{w;i}(r_i) = w_{r;i}R_i(r_i)$ where $\mathbf{w}_r$ is the n_r-vector of weights for the range variables and $w_{r;i}$ is its ith component. For convenience the weights are normalized so that $(\mathbf{1}^t\mathbf{w}_c)^2 + (\mathbf{1}^t\mathbf{w}_d)^2 + \mathbf{1}^t\mathbf{w}_r = 1$. With little additional complexity the dimension weights can be made dependent on the setting of the discrete variables but we will assume throughout that the weights are constant.

With these modifications, the total distance function becomes

$$d(\mathbf{x}, \boldsymbol{\varkappa}) = \left(\mathbf{x} - \boldsymbol{\mu}(\boldsymbol{\varkappa})\right)^t \mathbf{C}_w^{-1}(\boldsymbol{\varkappa})\left(\mathbf{x} - \boldsymbol{\mu}(\boldsymbol{\varkappa})\right) + Z_w(\boldsymbol{\varkappa}) + R_w(\mathbf{r}). \tag{1}$$

where $Z_w(\boldsymbol{\varkappa}) = \sum_{i=1}^{n_d} w_{d;i}\left\{w_{d;i}Z_i(\varkappa_i) + \sum_{j=1(\neq i)}^{n_d} w_{d;j}Z_{i,j}(\varkappa_i, \varkappa_j)\right\}$ and $R_w(\mathbf{r}) = \sum_{i=1}^{n_r} w_{r;i}R_i(r_i)$

Assigning weighting factors is useful only if the relevant contributions have been previously normalized so that they are all roughly the same magnitude. This serves as the baseline for which all weights are equal. The question immediately arises as to what criteria to use to weight the distance contributions.

We shall determine scaling factors, $Q_d > 0$ and $Q_r > 0$, so that the average distances per dimension of the discrete, range, and continuous contributions are equal, where by average we mean a utility-weighted average over the entire space of possible trades. This weighting places more emphasis on the better trades

If d_c, d_d, and d_r are the continuous, discrete, and range contributions to the total distance, then after multiplication by the scaling factors $d = d_c + Q_d d_d + Q_r d_r$. The scaling factors are determined through the utility weighted average distances defined by

$$\langle d_r \rangle \equiv \frac{\sum_{\boldsymbol{\varkappa}}\sum_r \int_V d\mathbf{u}\; Q_r d_r \exp[-Q_r d_r - Q_d d_d - d_c]}{\sum_{\boldsymbol{\varkappa}}\sum_r \int_V d\mathbf{u}\; \exp[-Q_r d_r - Q_d d_d - d_c]} = Q_r \frac{\sum_{\boldsymbol{\varkappa}} \exp[-Q_d d_d] \sum_r d_r \exp[-Q_r d_r] \int_V d\mathbf{u}\; \exp[-d_c]}{\sum_{\boldsymbol{\varkappa}} \exp[-Q_d d_d] \sum_r \exp[-Q_r d_r] \int_V d\mathbf{u}\; \exp[-d_c]}$$
$$\langle d_d \rangle \equiv \frac{\sum_{\boldsymbol{\varkappa}}\sum_r \int_V d\mathbf{u}\; Q_d d_d \exp[-Q_r d_r - Q_d d_d - d_c]}{\sum_{\boldsymbol{\varkappa}}\sum_r \int_V d\mathbf{u}\; \exp[-Q_r d_r - Q_d d_d - d_c]} = Q_d \frac{\sum_{\boldsymbol{\varkappa}} d_d \exp[-Q_d d_d] \sum_r \exp[-Q_r d_r] \int_V d\mathbf{u}\; \exp[-d_c]}{\sum_{\boldsymbol{\varkappa}} \exp[-Q_d d_d] \sum_r \exp[-Q_r d_r] \int_V d\mathbf{u}\; \exp[-d_c]}$$
$$\langle d_c \rangle \equiv \frac{\sum_{\boldsymbol{\varkappa}}\sum_r \int_V d\mathbf{u}\; d_c \exp[-Q_r d_r - Q_d d_d - d_c]}{\sum_{\mathbf{v}}\sum_r \int_V d\mathbf{u}\; \exp[-Q_r d_r - Q_d d_d - d_c]} = \frac{\sum_{\boldsymbol{\varkappa}} \exp[-Q_d d_d] \sum_r \exp[-Q_r d_r] \int_V d\mathbf{u}\; d_c \exp[-d_c]}{\sum_{\boldsymbol{\varkappa}} \exp[-Q_d d_d] \sum_r \exp[-Q_r d_r] \int_V d\mathbf{u}\; \exp[-d_c]}$$

A few comments on the above equations are in order. First, $\sum_{\boldsymbol{\varkappa}}$ indicates the repeated sum $\sum_{\varkappa_1} \cdots \sum_{\varkappa_{n_d}}$ over all possible discrete trades. $\sum_r$ indicates a sum over all the range variables and the integral over volume V indicates integration over the continuous trading volume of interest. Finally, we have not included a scaling factor Q_c on the continuous distance, since this can be made equal to 1 if we reinterpret Q_r as Q_r/Q_c and Q_d as Q_d/Q_c.[5] Each of the averages is an explicit function Q_d and Q_r.

The requirement on equal average contributions determines the two unknowns Q_r and Q_d through the equations: $\langle d_r \rangle / n_r = \langle d_c \rangle / n_c$ and $\langle d_d \rangle / n_d = \langle d_c \rangle / n_c$. These two nonlinear equations are coupled in terms of Q_r and Q_d and must be solved simultaneously for Q_r and Q_d. Further details are found in Appendix C.

[5] We singled the continuous variables out, since there will always be continuous variables in any trading scenario.

$\varkappa_1$	$\varkappa_2$	$\varkappa_3$
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

(a)

$\varkappa_2$
2

(b)

Table 2: An example set of constraints involving 3 variables where the domains of all variables are $\mathcal{D}_1 = \mathcal{D}_2 = \mathcal{D}_3 = \{1, 2, 3\}$. Constraint (a) requires that the values assumed by $\varkappa_1$, $\varkappa_2$, and $\varkappa_3$ are all different from each other, and constraint (b) requires that the value assumed by $\varkappa_2$ is even. See text for the solution to both constraints.

6.1.3 Constraint Specification

Buyers and sellers may express constraints over both continuous and discrete variables.

Continuous Constraints

For simplicity (and because additional expressiveness is rarely required) we assume that the buyer's constraints over the continuous variables are linear.[6] This allows a buyer to express a constraint, e.g., the time of delivery must be within 10 days or I will not trade, i.e., $t \leq 10$. We allow for both inequality and equality constraints which can be expressed as $\mathbf{G}_1^{(b)}\mathbf{x} = \mathbf{g}_1^{(b)}$ and $\mathbf{G}_2^{(b)}\mathbf{x} \leq \mathbf{g}_2^{(b)}$. If there are $c_1^{(b)}$ equality constraints then $\mathbf{G}_1^{(b)}$ has $c_1^{(b)}$ rows. Similarly, $\mathbf{G}_2^{(b)}$ has $c_2^{(b)}$ rows if there are $c_2^{(b)}$ inequality constraints. We allow the constraints to depend on the setting of the discrete variables, and to be explicit we often write $\mathbf{G}_1^{(b)}(\boldsymbol{\varkappa})$, $\mathbf{g}_1^{(b)}(\boldsymbol{\varkappa})$, $\mathbf{G}_2^{(b)}(\boldsymbol{\varkappa})$, and $\mathbf{g}_2^{(b)}(\boldsymbol{\varkappa})$.

Discrete Constraints

We use a standard methodology to represent and process constraints over discrete variables [5]. Abstractly, a constraint over a (perhaps proper)[7] subset of the discrete variables is represented as a list of all the allowed combinations the variables may assume. An example representation of a pair of discrete constraints is given in Table 2. There are two solutions to this set of constraints: $\varkappa_1 = 1$, $\varkappa_2 = 2$, $\varkappa_3 = 3$ and $\varkappa_1 = 3$, $\varkappa_2 = 2$, $\varkappa_3 = 1$. We indicate these solutions as $\{\langle\varkappa_1, 1\rangle, \langle\varkappa_2, 2\rangle\}, \langle\varkappa_3, 3\rangle\}\}$ and $\{\langle\varkappa_1, 3\rangle, \langle\varkappa_2, 2\rangle\}, \langle\varkappa_3, 1\rangle\}\}$ respectively. Each solution where all the variables have been identified with specific values from their domains is called a labelling.

[6] With little additional complexity we can also handle quadratic constraints, see [4].

[7] A proper subset of a set is the set itself.

Computationally efficient representations are used to ensure that only feasible combinations of variables are ever processed. Numerous third-party libraries offer constraint programming functionality.[8]

6.1.4 Utility and Total Cost of Ownership

The buyer's utility function and associated constraints may be difficult for many users to define. In this section we show how models of the buyer's business can be used to define utility in a natural manner.

We imagine a function which provides an estimate of the total cost of ownership for any given purchase. Cost contributions to this function might include piece part costs, freight costs, setup costs, quality assurance costs, repair costs, etc. It is important to include all quantifiable costs associated with the lifetime of use of the purchased product because it is this function we will be minimizing. Significant savings may be obtained by taking a longer-term view of the purchase. Revenues (negative costs) generated from the purchase are also included in the function so that the function represents some measure of profitability associated with the purchase. We write the total cost of ownership function as $C_o(\mathbf{x}, \boldsymbol{\varkappa}, \mathbf{r}; \boldsymbol{\beta})$. We explicitly indicate the dependence on the negotiated trade parameters $\mathbf{x}$, $\boldsymbol{\varkappa}$, and $\mathbf{r}$, as well as other factors $\boldsymbol{\beta}$. The other factors might include forecasted demand, current inventory levels, etc. These factors will vary over time, and they can be extracted from the buyer's ERP and supply chain management systems (SCM) in real-time just before the purchase to ensure continuous real-time optimization. See section 6.2.1 for further details.

Minimization of $C_o(\mathbf{x}, \boldsymbol{\varkappa}, \mathbf{r}; \boldsymbol{\beta})$ defines an ideal trade dependent on current conditions: $\mathbf{x}_{\text{opt}}(\boldsymbol{\beta})$, $\boldsymbol{\varkappa}_{\text{opt}}(\boldsymbol{\beta})$, $\mathbf{r}_{\text{opt}}(\boldsymbol{\beta})$. If desired, these can be used to define $\boldsymbol{\mu} = \mathbf{x}_{\text{opt}}(\boldsymbol{\beta}$, $r = \mathbf{r}_{\text{opt}}(\boldsymbol{\beta})$ and the desired ideal discrete configuration $\boldsymbol{\varkappa}_{\text{opt}}(\boldsymbol{\beta})$ (having distance contribution $Z = 0$). Moreover, the tradeoffs between continuous dimensions around this minimum can be obtained through calculation of the Hessian matrix $\mathbf{H} = [h_{i,j}]$ where the i, j matrix element is given by

$$h_{i,j} = \left. \frac{\partial^2 C_o(\mathbf{x}, \boldsymbol{\varkappa}, \mathbf{r}; \boldsymbol{\beta})}{\partial x_i \partial x_j} \right|_{\mathbf{x}=\mathbf{x}_{\text{opt}}(\boldsymbol{\beta}), \boldsymbol{\varkappa}=\boldsymbol{\varkappa}_{\text{opt}}(\boldsymbol{\beta}), \mathbf{r}=\mathbf{r}_{\text{opt}}(\boldsymbol{\beta})}$$

We then identify $\mathbf{C}^{-1}$ with $\mathbf{H}$. In this way, little trading flexibility is obtained in directions where total cost of ownership rises rapidly, while significant flexibility is obtained in directions where total cost of ownership increases slowly.

In summary, a total cost of ownership model defines both the most preferred trade parameters and the flexibility possessed around the preferred trade. The model pulls dynamically from real-time data sources to provide the most up-to-date optimization based on total costs of ownership and other important qualitative factors the buyer may wish to describe in the utility function. The same function and its constituent costs may also be used to help analyze proposed trades from suppliers.

[8] See for example `www.mozart-oz.org` or `www.ilog.com`.

6.1.5 Supplier Capabilities

As discussed in the introduction, suppliers represent their capabilities through a specification of the subspace of X in which they will trade. A supplier's capabilities must specify the allowed continuous, discrete, and range variables. The allowed range variables are expressed as the pairs $(\underline{r}_j, \overline{r}_j)$, one for each range variable. For example, if a supplier produces 25mm inner diameter screws to within a tolerance of 0.5 mm, then the range variable is simply $(24.5, 25.5)$. These are compared with the buyer's ideal range and contribute to the distance function through the $\mathbf{R}_w(\varkappa)$ function.

Capabilities over continuous and discrete variables are more complex.

Continuous Capabilities

Continuous capabilities are viewed naturally as responses to a buyer's request. Thus we distinguish between a buyer's requested continuous vector $\mathbf{x}^{(b)}$ and a seller's response $\mathbf{x}^{(s)}$. A vector-valued function, $\mathbf{f}(\mathbf{x}^{(b)}, \mathbf{x}^{(s)}, \varkappa)$ returns the response based on the buyer's request and also, perhaps, other previously defined supplier responses. Component f_i of $\mathbf{f}$ defines the ith continuous variable, *i.e.* $x_i^{(s)} = f_i(\mathbf{x}^{(b)}, \mathbf{x}^{(s)})$.

Currently, suppliers are used to quoting price discounts for large volume orders and these price discounts are expressed as piecewise linear functions. Consequently, we restrict f_i to have the following form (where we distinguish between the functions depending on the buyer and seller variables):

$$x_i^{(s)} = \sum_k f_{i,k}^{(s)}(x_k^{(s)}, \varkappa^{(s)}) + f_{i,k}^{(b)}(x_k^{(b)}, \varkappa^{(s)}). \tag{2}$$

An example of how this may be used to define a supplier response is the following: We assume three continuous dimensions – price, volume, and time of delivery and indicate these as $[x_1, x_2, x_3] = [p, v, t]$. Then a response may be formed as

$$\begin{aligned}
v^{(s)} &= f_{v,v}(v^{(b)}) \\
t^{(s)} &= f_t(v^{(s)}, t^{(b)}) = f_{t,v}(v^{(s)}) \\
p^{(s)} &= f_p(v^{(s)}, t^{(s)}) = f_{p,v}(v^{(s)}) + f_{p,t}(t^{(s)}).
\end{aligned}$$

The $f_{v,v}$ function returns the volume a supplier will fulfill as a function of what the buyer asked for. If the supplier can deliver any volume, this will be the identity function. If the supplier delivers only in certain lot sizes, this function may have a staircase shape, etc. The $f_{t,v}$ function indicates the time it will take a supplier to deliver a certain volume. So, for example, if larger shipments require longer transportation, then this dependence is given by this function. Finally, we turn to the price determination. In this example the price depends on the quantity $v^{(s)}$ being shipped and the $f_{p,v}$ might represent price discounts for large volume orders. There is also an incremental price contribution based on the time of delivery. If faster delivery is more expensive, this is reflected in $f_{p,t}$.

For a given setting of the discrete variables, each $f_{i,k}^{(s)}(x_k^{(s)}, \boldsymbol{\varkappa}^{(s)})$ and $f_{i,k}^{(b)}(x_k^{(b)}, \boldsymbol{\varkappa}^{(s)})$is a one-dimensional piecewise linear function. Consequently, the functions can be specified by listing the breakpoints. If $f_{i,k}^{(s)}(x_k^{(s)}, \boldsymbol{\varkappa}^{(s)})$ has $\kappa_{i,k}^{(s)}$ breakpoints, then we list these as $\{x_k^{(s)}(b_{i,k}^{(s)} = 1), x_k^{(s)}(b_{i,k}^{(s)} = 2), \cdots, x_k^{(s)}(b_{i,k}^{(s)} = \kappa_{i,k}^{(s)})\}$[9] and function values at these breakpoints are $\{f_{i,k}^{(s)}(x_k^{(s)}(b_{i,k}^{(s)} = 1)), f_{i,k}^{(s)}(x_k^{(s)}(b_{i,k}^{(s)} = 2)), \cdots, f_{i,k}^{(s)}(x_k^{(s)}(b_{i,k}^{(s)} = \kappa_{i,k}^{(s)}))\}$. Similarly, $f_{i,k}^{(b)}$, is a piecewise linear function defined by the $\kappa_{i,k}^{(b)}$ breakpoints $\{x_k^{(b)}(b_{i,k}^{(b)} = 1), x_k^{(b)}(b_{i,k}^{(b)} = 2), \cdots, x_k^{(b)}(b_{i,k}^{(b)} = \kappa_{i,k}^{(b)})\}$ and function values at these breakpoints $\{f_{i,k}^{(b)}(x_k^{(b)}(b_{i,k}^{(b)} = 1)), f_{i,k}^{(b)}(x_k^{(b)}(b_{i,k}^{(b)} = 2)), \cdots, f_{i,k}^{(b)}(x_k^{(b)}(b_{i,k}^{(b)} = \kappa_{i,k}^{(b)}))\}$. The breakpoints are indexed by the integers $b_{i,k}^{(s)}$ and $b_{i,k}^{(b)}$.

An interval is specified by assigning a value $b_{i,k}^{(s)} \in [1, \kappa_{i,k}^{(s)} - 1]$ and $b_{i,k}^{(b)} \in [1, \kappa_{i,k}^{(b)} - 1]$ so that[10]

$$x_k^{(s)}(b_{i,k}^{(s)}) \le x_k^{(s)} \le x_k^{(s)}(b_{i,k}^{(s)} + 1) \;\; \forall\, i \quad \text{and} \quad x_k^{(b)}(b_{i,k}^{(b)}) \le x_k^{(b)} \le x_k^{(b)}(b_{i,k}^{(b)} + 1) \;\; \forall\, i. \tag{3}$$

Within each interval the functions are linear, so we have

$$x_i^{(s)} = c_i(\boldsymbol{\varkappa}^{(s)}, \{b_{i,k}^{(s)}\}, \{b_{i,k}^{(b)}\}) + \sum_k m_{i,k}^{(s)}(\boldsymbol{\varkappa}^{(s)}, b_{i,k}^{(s)}) x_k^{(s)} + m_{i,k}^{(b)}(b_{i,k}^{(b)}) x_k^{(b)}$$

where $c_i(\mathbf{v}^{(s)}, \{b_{i,k}^{(s)}\}, \{b_{i,k}^{(b)}\}) = \sum_k c_{i,k}^{(s)}(\boldsymbol{\varkappa}^{(s)}, b_{i,k}^{(s)}) + c_{i,k}^{(b)}(b_{i,k}^{(b)})$. In the above equations, the intercepts and slopes are given explicitly by

$$c_{i,k}^{(s)}(b_{i,k}^{(s)}) = \frac{x_k^{(s)}(b_{i,k}^{(s)} + 1) f_{i,k}^{(s)}(x_k^{(s)}(b_{i,k})) - x_k^{(s)}(b_{i,k}) f_{i,k}^{(s)}(x_k^{(s)}(b_{i,k} + 1))}{x_k^{(s)}(b_{i,k}^{(s)} + 1) - x_k^{(s)}(b_{i,k})}$$

and

$$m_{i,k}^{(s)}(b_{i,k}^{(s)}) = \frac{f_{i,k}^{(s)}(x_k^{(s)}(b_{i,k}^{(s)} + 1)) - f_{i,k}^{(s)}(x_k^{(s)}(b_{i,k}))}{x_k^{(s)}(b_{i,k}^{(s)} + 1) - x_k^{(s)}(b_{i,k}^{(s)})}$$

respectively. An analogous result holds for the $c_{i,k}^{(b)}(b_{i,k}^{(b)})$ and $m_{i,k}^{(b)}(b_{i,k}^{(b)})$.

To eliminate any cyclic dependence on $x_i^{(s)}$ we must impose an ordering on $x_i^{(s)}$ so that $x_i^{(s)}$ can only depend on $x_j^{(s)}$ where $j < i$. Consequently, we can write

$$x_i^{(s)} = c_i(\boldsymbol{\varkappa}^{(s)}, \{b_{i,1}^{(s)}, \cdots, b_{i,i-1}^{(s)}\}, \{b_{i,k}^{(b)}\}) + \sum_{k<i} m_{i,k}^{(s)}(\boldsymbol{\varkappa}^{(s)}, b_{i,k}^{(s)}) x_k^{(s)} + \sum_k m_{i,k}^{(b)}(b_{i,k}^{(b)}) x_k^{(b)}.$$

Written as a matrix equation, the above becomes

$$\left(\mathbf{I} - \mathbf{M}^{(s)}(\boldsymbol{\varkappa}^{(s)}, \{b_{i,k}^{(s)}\})\right)\mathbf{x}^{(s)} = \mathbf{c}(\boldsymbol{\varkappa}^{(s)}, \{b_{i,k}^{(s)}\}, \{b_{i,k}^{(b)}\}) + \mathbf{M}^{(b)}(\{b_{i,k}^{(b)}\})\mathbf{x}^{(b)}$$

where $\mathbf{c}(\boldsymbol{\varkappa}^{(s)}, \{b_{i,k}^{(s)}\}, \{b_{i,k}^{(b)}\}) = \begin{bmatrix} c_1(\boldsymbol{\varkappa}^{(s)}, \{b_{i,k}^{(s)}\}, \{b_{i,k}^{(b)}\}) & \cdots & c_n(\boldsymbol{\varkappa}^{(s)}, \{b_{i,k}^{(s)}\}, \{b_{i,k}^{(b)}\}) \end{bmatrix}^t$, $\mathbf{x}^{(s)} = \begin{bmatrix} x_1^{(s)} & \cdots & x_{n_c}^{(s)} \end{bmatrix}^t$,

[9]The breakpoints are assumed to be in increasing order.

[10]Note that many choices for $\{b_{i,k}^{(s)}\}$ or $\{b_{i,k}^{(b)}\}$ would be inconsistent with these constraints. A simple way to fix this is to define a union of all breakpoints (s) and (b), and have all $f_{i,k}$ have the same breakpoints.

$\mathbf{x}^{(b)} = \begin{bmatrix} x_1^{(b)} & \cdots & x_{n_c}^{(b)} \end{bmatrix}^t$, and

$$\mathbf{M}^{(s)}(\boldsymbol{\varkappa}^{(s)}, \{b_{i,k}^{(s)}\}) = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ m_{2,1}^{(s)}(\boldsymbol{\varkappa}^{(s)}, b_{2,1}^{(s)}) & 0 & \cdots & 0 & 0 \\ m_{3,1}^{(s)}(\boldsymbol{\varkappa}^{(s)}, b_{3,1}^{(s)}) & m_{3,2}^{(s)}(\boldsymbol{\varkappa}^{(s)}, b_{3,2}^{(s)}) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ m_{n,1}^{(s)}(\boldsymbol{\varkappa}^{(s)}, b_{n,1}^{(s)}) & m_{n,2}^{(s)}(\boldsymbol{\varkappa}^{(s)}, b_{n,2}^{(s)}) & \cdots & m_{n,n-1}^{(s)}(\boldsymbol{\varkappa}^{(s)}, b_{n,n-1}^{(s)}) & 0 \end{bmatrix}$$

and

$$\mathbf{M}^{(b)}(\{b_{i,k}^{(b)}\}) = \begin{bmatrix} m_{1,1}^{(b)}(b_{1,1}^{(b)}) & m_{1,2}^{(b)}(b_{1,2}^{(b)}) & \cdots & m_{1,n}^{(b)}(b_{1,n}^{(b)}) \\ m_{2,1}^{(b)}(b_{2,1}^{(b)}) & m_{2,2}^{(b)}(b_{2,2}^{(b)}) & \cdots & m_{2,n}^{(b)}(b_{2,n}^{(b)}) \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1}^{(b)}(b_{n,1}^{(b)}) & m_{n,2}^{(b)}(b_{n,2}^{(b)}) & \cdots & m_{n,n}^{(b)}(b_{n,n}^{(b)}) \end{bmatrix}.$$

In most cases $\mathbf{x}^{(s)}$ will depend only on a subset of the variables in $\mathbf{x}^{(b)}$. If $\mathbf{x}^{(s)}$ depends on $n' \leq n$ of the $\mathbf{x}^{(b)}$ variables, then $\mathbf{M}^{(b)}$ is an $n \times n'$ matrix. In the example given everything depending only upon the volume the buyer requested.

Since $\mathbf{M}^{(s)}(\boldsymbol{\varkappa}^{(s)})$ is lower triangular and can be inverted in time $\mathcal{O}(n)$, we can rapidly express $\mathbf{x}^{(s)}$ as

$$\mathbf{x}^{(s)} = \left(\mathbf{I} - \mathbf{M}^{(s)}(\boldsymbol{\varkappa}^{(s)}, \{b_{i,k}^{(s)}\})\right)^{-1}\mathbf{c}(\boldsymbol{\varkappa}^{(s)}, \{b_{i,k}^{(s)}\}, \{b_{i,k}^{(b)}\}) + \left(\mathbf{I} - \mathbf{M}^{(s)}(\boldsymbol{\varkappa}^{(s)}, \{b_{i,k}^{(s)}\})\right)^{-1}\mathbf{M}^{(b)}(\{b_{i,k}^{(b)}\})\mathbf{x}^{(b)} \quad (4)$$

as long as the $b_{i,k}^{(s)}$ are chosen to also satisfy $x_k^{(s)}(b_{i,k}^{(s)}) \leq x_k^{(s)} \leq x_k^{(s)}(b_{i,k}^{(s)} + 1)$. These constraints will be used in section 6.1.6 which formulates the optimization problem.

We also allow a supplier to express additional linear constraints so that, for example, he may represent that he does not deliver on Sunday. Thus the supplier may define the matrices $\mathbf{G}_1^{(s)}(\boldsymbol{\varkappa})$, $\mathbf{G}_2^{(s)}(\boldsymbol{\varkappa})$, and the vectors $\mathbf{g}_1^{(s)}(\boldsymbol{\varkappa})$, $\mathbf{g}_2^{(s)}(\boldsymbol{\varkappa})$ such that $\mathbf{G}_1^{(s)}\mathbf{x}^{(s)} = \mathbf{g}_1^{(s)}$ and $\mathbf{G}_2^{(s)}\mathbf{x}^{(s)}\mathbf{x} \leq \mathbf{g}_2^{(s)}$. $\mathbf{G}_1^{(s)}(\boldsymbol{\varkappa})$ and $\mathbf{G}_2^{(s)}(\boldsymbol{\varkappa})$ have $c_1^{(s)}$ and $c_2^{(s)}$ rows respectively.

Discrete Capabilities

It is easy to imagine that a supplier's response on a discrete dimension is highly constrained by the values of the response on other dimensions, e.g., certain product characteristics come only in certain colors and package sizes. Consequently, it is not suitable to explicitly define a response but only to make available a supplier's constraints amongst the discrete variables. Consider 3 discrete dimensions where $\varkappa_1 \in \mathcal{D}_1 = [a, b, c]$, $\varkappa_2 \in \mathcal{D}_2 = [A, B, C, D]$, and $\varkappa_3 \in \mathcal{D}_3 = [\alpha, \beta, \gamma, \delta]$, and assume the supplier has the following 3 constraints

$$\mathcal{C}_1(\varkappa_1, \varkappa_3) = \{(a, \alpha), (a, \beta), (b, \beta), (c, \beta)\},$$
$$\mathcal{C}_2(\varkappa_2, \varkappa_3) = \{(A, \beta), (B, \gamma), (D, \beta)\},$$
$$\mathcal{C}_3(\varkappa_1) = \{b, c\}.$$

We first note that there are 4 feasible solutions (or product configurations the supplier can meet): $[\varkappa_1, \varkappa_2, \varkappa_3] = [b, A, \beta]$, $[b, D, \beta]$, $[c, A, \beta]$, or $[c, D, \beta]$. Feasible solutions to the constraints define the response $\boldsymbol{\varkappa}^{(s)}$ for the discrete variables.

We indicate a supplier's or buyer's collective set of discrete constraints by $\mathcal{C}^{(s)}(\boldsymbol{\varkappa})$ and $\mathcal{C}^{(b)}(\boldsymbol{\varkappa})$ respectively.

6.1.6 The Optimization Problem

Having defined the necessary components, we now define the optimization task which determines the continuous $\mathbf{x}^*$ and discrete $\boldsymbol{\varkappa}^*$ parameters of the trade.

Since the trade must be acceptable to the supplier, we maximize the buyer's utility over a supplier's capabilities. Equivalently, we minimize the distance from the buyer's ideal values as

$$[\mathbf{x}^*, \boldsymbol{\varkappa}^*] = \arg\min_{\mathbf{x}^{(s)}, \boldsymbol{\varkappa}^{(s)}} \left\{ \left(\mathbf{x}^{(\mathbf{s})} - \boldsymbol{\mu}(\boldsymbol{\varkappa}^{(s)})\right)^t \mathbf{C}_w^{-1}(\boldsymbol{\varkappa}^{(s)}) \left(\mathbf{x}^{(\mathbf{s})} - \boldsymbol{\mu}(\boldsymbol{\varkappa}^{(s)})\right) + Q_d Z_w(\boldsymbol{\varkappa}^{(s)}) + Q_r R_w(\mathbf{w}) \right\}$$

where

$$\mathbf{x}^{(s)} = \left(\mathbf{I} - \mathbf{M}^{(s)}(\boldsymbol{\varkappa}^{(s)})\right)^{-1} \mathbf{c}(\boldsymbol{\varkappa}^{(s)}) + \left(\mathbf{I} - \mathbf{M}^{(s)}(\boldsymbol{\varkappa}^{(s)})\right)^{-1} \mathbf{M}^{(b)} \mathbf{x}^{(b)}$$

subject to the constraints over continuous variables

$$\mathbf{G}_1(\boldsymbol{\varkappa}^{(s)}) \mathbf{x}^{(s)} = \mathbf{g}_1(\boldsymbol{\varkappa}^{(s)}), \qquad \mathbf{G}_2(\boldsymbol{\varkappa}^{(s)}) \mathbf{x}^{(s)} \leq \mathbf{g}_2(\boldsymbol{\varkappa}^{(s)})$$

and the constraints over the discrete variables $\mathcal{C}^{(b)}(\mathbf{v}^{(s)})$, $\mathcal{C}^{(s)}(\mathbf{v}^{(s)})$. In the above, we have defined the $(c_1^{(s)} + c_1^{(b)}) \times n_c$ and $(c_2^{(s)} + c_2^{(b)}) \times n_c$ matrices $\mathbf{G}_1(\boldsymbol{\varkappa}^{(s)})$ and $\mathbf{G}_2(\boldsymbol{\varkappa}^{(s)})$ by

$$\mathbf{G}_1(\boldsymbol{\varkappa}^{(s)}) = \begin{bmatrix} \mathbf{G}_1^{(s)}(\boldsymbol{\varkappa}^{(s)}) \\ \mathbf{G}_1^{(b)}(\boldsymbol{\varkappa}^{(s)}) \end{bmatrix}, \quad \text{and} \quad \mathbf{G}_2(\boldsymbol{\varkappa}^{(s)}) = \begin{bmatrix} \mathbf{G}_2^{(s)}(\boldsymbol{\varkappa}^{(s)}) \\ \mathbf{G}_2^{(b)}(\boldsymbol{\varkappa}^{(s)}) \end{bmatrix}.$$

The $(c_1^{(s)} + c_1^{(b)})$- and $(c_1^{(s)} + c_1^{(b)})$-vectors $\mathbf{g}_1(\boldsymbol{\varkappa}^{(s)})$ and $\mathbf{g}_2(\boldsymbol{\varkappa}^{(s)})$ are defined by

$$\mathbf{g}_1(\boldsymbol{\varkappa}^{(s)}) = \begin{bmatrix} \mathbf{g}_1^{(s)}(\boldsymbol{\varkappa}^{(s)}) \\ \mathbf{g}_1^{(b)}(\boldsymbol{\varkappa}^{(s)}) \end{bmatrix}, \quad \text{and} \quad \mathbf{g}_2(\boldsymbol{\varkappa}^{(s)}) = \begin{bmatrix} \mathbf{g}_2^{(s)}(\boldsymbol{\varkappa}^{(s)}) \\ \mathbf{g}_2^{(b)}(\boldsymbol{\varkappa}^{(s)}) \end{bmatrix}.$$

The optimization is accomplished by iterating two distinct phases. Phase one sets the continuous parameters optimally for a given setting of the discrete variables. We define the functions

$$d_1(\mathbf{x}, \boldsymbol{\varkappa}) = \left(\mathbf{x} - \boldsymbol{\mu}(\boldsymbol{\varkappa})\right)^t \mathbf{C}_w^{-1}(\boldsymbol{\varkappa})\left(\mathbf{x} - \boldsymbol{\mu}(\boldsymbol{\varkappa})\right) + R(\mathbf{r}; \boldsymbol{\varkappa}) \quad \text{and} \quad d_2(\boldsymbol{\varkappa}) = Q_d Z_w(\boldsymbol{\varkappa}^{(s)}),$$

The first phase of the optimization is the continuous problem:[11]

$$\mathbf{x}^*(\boldsymbol{\varkappa}) = \arg\min_{\mathbf{x}^{(s)}} d_1(\mathbf{x}^{(s)}, \boldsymbol{\varkappa}) \quad \text{subject to} \quad \mathbf{G}_1(\boldsymbol{\varkappa})\mathbf{x} = \mathbf{g}_1(\boldsymbol{\varkappa}) \text{ and } \mathbf{G}_2(\boldsymbol{\varkappa})\mathbf{x} \leq \mathbf{g}_2(\boldsymbol{\varkappa}). \tag{5}$$

[11] No optimization is required over range variables, since these are specified up front by both buyer and seller and merely add to the total distance.

A detailed discussion on the solution of the phase 1 optimization problem is found in appendix D. The second phase determines the best value of the discrete variables by minimizing over a function of $\varkappa$ alone

$$\varkappa^* = \arg\min_{\varkappa}\; d_1(\mathrm{x}(\varkappa), \varkappa) + d_2(\varkappa) \quad \text{subject to} \quad \mathcal{C}^{(b)}(\varkappa) \wedge \mathcal{C}^{(s)}(\varkappa). \tag{6}$$

Further details on the phase 2 optimization are given in Appendix E. Once $\varkappa^*$ has been determined, we find $\mathbf{x}^*$ as $\mathbf{x}^* = \mathbf{x}(\varkappa^*)$.

6.1.7 Aggregation

Often a buyer may be willing to divide an order between multiple suppliers in order to aggregate the required demand or to obtain better deals. In this section, we detail how the present invention supports this aggregate optimization.

Aggregation can only occur over the continuous variables where values may be subdivided. Each continuous variable x_i must be parcelled out amongst a set of suppliers. Consequently, we extend our notation to $x_i \to \tilde{x}_{i,\,k}$ giving the contribution of the kth supplier to continuous dimension i. The kth supplier may come from a (perhaps proper) subset of all suppliers. We indicate the set of potentially contributing suppliers as $\mathcal{K}$ and the number of potentially contributing suppliers as $|\mathcal{K}|$.[12]

We restrict the discrete variables to be the same across all potentially aggregated suppliers, i.e., we do *not* generalize $\varkappa_i \to \varkappa_{i,k}$. This simplifying assumption is made for two reasons. First, the size of the discrete optimization problem is smaller and so optimization be performed faster. Second, it may be difficult to elicit from the buyer the allowed discrete alternatives for each supplier. Nevertheless, this generalization is straightforward should the need arise.

This simplifying assumption requires that the union of discrete supplier constraints $\mathcal{C}_{\mathcal{K}}(\varkappa) \equiv \bigwedge_{k\in\mathcal{K}} \mathcal{C}_k^{(s)}(\varkappa)$ yields a feasible solution when combined with the buyer's discrete constraints $\mathcal{C}^{(b)}(\varkappa)$. A necessary (but not sufficient) condition for satisfaction is then that each constraint satisfaction problem k having constraints $\mathcal{C}^{(b)}(\varkappa) \wedge \mathcal{C}_k^{(s)}(\varkappa)$ has a feasible solution.[13] Henceforth, we will assume that the set of suppliers, $\mathcal{K}$, satisfies this condition. If not, those suppliers violating the constraints $\mathcal{C}^{(b)}(\varkappa) \wedge \mathcal{C}_k^{(s)}(\varkappa)$ are eliminated from consideration in $\mathcal{K}$.

Discrete Search

We must search over the subsets of $\mathcal{K}$ for feasible solutions, which is a combinatorial problem. Fortunately, given a complete labelling of variables, determining the largest subset is easy. For any given labelling of all discrete variables, if each $\mathcal{C}^{(b)} \wedge \mathcal{C}_k^{(s)}\ \forall\ k \in \kappa \subset \mathcal{K}$ is satisfiable, then the union $\mathcal{C}^{(b)} \wedge \mathcal{C}_\kappa^{(s)}$ where $\mathcal{C}_\kappa^{(s)} = \bigwedge_{k\in\kappa} \mathcal{C}_k^{(s)}$ is also satisfiable under the same labelling. The largest subset of variables is found by adding all k which

[12] All work to this point is thus seen as the special case $|\mathcal{K}| = 1$.

[13] It is easily seen this requirement is not sufficient by having $\mathcal{C}^{(b)} = \{\langle\varkappa_1, 1\rangle, \langle\varkappa_1, 2\rangle\}$, $\mathcal{C}_1^{(s)} = \{\langle\varkappa_1, 1\rangle\}$, and $\mathcal{C}_2^{(s)} = \{\langle\varkappa_1, 2\rangle\}$.

have feasible solutions with the buyer. We needn't worry about smaller subsets because the continuous optimization will assign zero values to those if appropriate. Consequently, for any given labelling $\boldsymbol{\varkappa}$ we let $\mathcal{K}(\boldsymbol{\varkappa})$ represent the maximal subset of suppliers for which $\mathcal{C}^{(b)}(\boldsymbol{\varkappa}) \wedge \mathcal{C}_{\mathcal{K}}(\boldsymbol{\varkappa})$ is satisfiable. It is this set of suppliers which enter into the continuous optimization. The number of participating suppliers is denoted by $|\mathcal{K}(\boldsymbol{\varkappa})|$.

Continuous Optimization

Optimization over the continuous variables is carried for each labelling $\boldsymbol{\varkappa}$. Generally speaking, the buyer's utility will not be an explicit function of $x_{i;k}$ but only of x_i. We assume a linear relationship between these two quantities so that[14]

$$\mathbf{x} = \Xi \tilde{\mathbf{x}}.$$

The $n_c|\mathcal{K}(\boldsymbol{\varkappa})|$ vector $\tilde{x}$ is defined as $\tilde{\mathbf{x}}^t = [\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_{n_c}]$ where $\tilde{\mathbf{x}}_i^t = [\tilde{x}_{i;1}, \cdots, \tilde{x}_{i;|\mathcal{K}(\boldsymbol{\varkappa})|}]$. The $n_c \times n_c|\mathcal{K}(\boldsymbol{\varkappa})|$ matrix Ξ $\xi_{i,k}$ is assumed known and typically has the form[15]

$$\Xi = \begin{bmatrix} \boldsymbol{\xi}_1^t & \mathbf{0}^t & \cdots & \mathbf{0}^t \\ \mathbf{0}^t & \boldsymbol{\xi}_2^t & \cdots & \mathbf{0}^t \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}^t & \mathbf{0}^t & \cdots & \boldsymbol{\xi}_{n_c}^t \end{bmatrix}$$

where $\mathbf{0}$ is the K-vector of all zeros and $\boldsymbol{\xi}_i$ is the linear combination relating x_i to the $\tilde{x}_{i;k}$. Under our assumptions for Ξ, $x_i = \boldsymbol{\xi}_i^t \tilde{\mathbf{x}}_i$. In cases where the buyer wants to accumulate the results from suppliers (e.g., aggregating quantities) $\boldsymbol{\xi} = \mathbf{1}$ is the $|\mathcal{K}(\boldsymbol{\varkappa})|$-vector of all 1s. In other cases the buyer may take $\boldsymbol{\xi} = \mathbf{1}/|\mathcal{K}(\boldsymbol{\varkappa})|$ so that the time of delivery becomes the average time of delivery across the suppliers. Constraints on $\mathbf{x}$ become constraints on $\tilde{\mathbf{x}}$ via

$$\tilde{\mathbf{G}}_1(\boldsymbol{\varkappa})\tilde{\mathbf{x}} = \mathbf{g}_1(\boldsymbol{\varkappa}) \quad \text{and} \quad \tilde{\mathbf{G}}_2(\boldsymbol{\varkappa})\tilde{\mathbf{x}} \leq \mathbf{g}_2(\boldsymbol{\varkappa}) \tag{7}$$

where $\tilde{\mathbf{G}}_1(\boldsymbol{\varkappa}) = \tilde{\mathbf{G}}_1(\boldsymbol{\varkappa})\Xi$ and $\tilde{\mathbf{G}}_1(\boldsymbol{\varkappa}) = \tilde{\mathbf{G}}_1(\boldsymbol{\varkappa})\Xi$. We might also expect the buyer to add additional linear constraints, such as requiring the latest shipment from any supplier to arrive earlier than a certain date, or requiring all deliveries to arrive the same day. There can also be constraints specific to particular suppliers, e.g., the buyer doesn't want any more than 100 units from supplier 5. These can be handled simply as constraints on the individual $\tilde{x}_{i;k}$ and added as extra rows to $\tilde{\mathbf{G}}_1(\boldsymbol{\varkappa})$, $\tilde{\mathbf{G}}_2(\boldsymbol{\varkappa})$, $\tilde{\mathbf{g}}_1(\boldsymbol{\varkappa})$, and $\tilde{\mathbf{g}}_2(\boldsymbol{\varkappa})$.

With aggregation, the quadratic form to be minimized is $\big(\Xi\tilde{\mathbf{x}} - \boldsymbol{\mu}(\boldsymbol{\varkappa})\big)^t \mathbf{C}_w^{-1}(\boldsymbol{\varkappa})\big(\Xi\tilde{\mathbf{x}} - \boldsymbol{\mu}(\boldsymbol{\varkappa})\big)$ subject to the constraints given in Eq. (7). This minimization can be carried out through a straightforward generalization of the method given in Appendix D.

[14] We can also relate $\mathbf{x}$ and $\tilde{\mathbf{x}}$ by $\mathbf{x} = \Xi\tilde{\mathbf{x}} + \boldsymbol{\nu}$ for some constant vector $\boldsymbol{\nu}$, which will not cause any complications. However, there seems to be little practical reason to do so.

[15] With no additional computational complexity, we can allow Ξ to depend on $\boldsymbol{\varkappa}$.

6.2 Implementation

In this section we outline an implementation of the entire e-procurement invention. We begin with a high-level description of the architecture, then fill in the details by describing a complete object model.

6.2.1 High-level Architecture of the Invention

There are at least two modes in which the invention may be used. First, the invention may reside at the site of large buyers, and suppliers who wish to sell to the buyer may be required to submit their capabilities via a web interface to the buyer. The invention may also be used within a marketplace hosted by a third party. Buyers/sellers log onto the market, submit their preference/capabilities, and act on the results. The architecture is modular enough to support both modes of operation.

In Figure 1 we present an architecture for the invention. We describe the architecture, starting from the optimization algorithm which finds matches between buyers and sellers and work our way outwards.

A controller surrounds the optimization engine, feeding it buyer preferences and seller capabilities. If multiple optimization processes are running (perhaps on different machines), the controller can also do load balancing, forwarding the request to the least busy process. The controller decomposes preferences and capabilities into their constituent buyer- and seller-specific versions (see below), selects the most specific matching preference/capability pairs, and sends them to the matching engine for optimization. The controller then collects responses from the matching engine and returns them to the buyer. Additionally, the controller logs all results into a database for recording purposes.

Another layer, called the Connector in Figure 1, separates the graphical user interface (GUI) through which users communicate with the tool from the controller. This layer serves a number of functions. The connector transforms the description of preferences and capabilities from the GUI into a form suitable for the implementation of the matching engine. Part of this transformation involves validation of appropriate input from the GUI layer so that no malformed input is ever sent to the controller. The Connector layer can also pull data from ERP or SCM systems and automatically infer preferences (using the total cost of ownership function) for the buyer. The enterprise abstraction layer insulates the Connector from the precise details of the manner in which the ERP and SCM data needs to be gathered. Total cost of ownership is evaluated in the simulation modules, which may either be running locally at the client's site or running centrally at the main server. These simulation modules pull operational data (the vector β)from the enterprise abstraction layer. A preference optimization module (TCO) minimizes the total cost of ownership to determine the ideal trade and the flexibilities around the ideal trade.

At the outmost level, a layer provides integration with the GUI and/or host system. A number of administrative systems are expected at this layer. Market administration services allow easy definition of trading spaces, the dimensions of negotiation, limits on continuous variables, allowed settings of the

discrete variables, etc. User administration services allow an administrator to define buyers, passwords, spending limits, etc. Supplier services accomplish similar tasks on the supply side. Managers for preferences, capabilities, and match results ensure that these objects are properly stored in a database. This layer layer also dynamically generates the html necessary for presentation of the data via a web interface to buyers and sellers.

For maximal portability, communications between the View and Connector are via XML documents. For maximal efficiency, communications between the Connector and matching controller are as serialized Java objects.

6.2.2 An Object Model for the Invention

The fundamental objects required for the invention are preferences from buyers, capabilities from sellers, and match results returned to all parties. The components of such objects have already been considered from a mathematical point of view, and we now describe one possible computer representation.

In this section we describe a complete grammar for the object model. The following syntactic conventions are used:

- ⟨nt⟩ denotes a non-terminal symbol nt
- [obj] denotes an optional grammar segment obj
- {obj} denotes 1, or many times the grammar segment obj
- → denotes a production rule for non-terminal symbol. If there are multiple rules, say ⟨a⟩, ⟨b⟩, and ⟨c⟩, then these are denoted as

  $$\langle \text{nt} \rangle \;\rightarrow \langle a \rangle | \langle b \rangle | \langle c \rangle.$$

  In contrast, a production rule of the form

  $$\langle \text{nt} \rangle \;\rightarrow \langle a \rangle, \langle b \rangle, \langle c \rangle$$

  indicates that the non-terminal ⟨nt⟩ is composed of three grammar segments, ⟨a⟩, ⟨b⟩, and ⟨c⟩
- terminal keywords are in serif font

Obvious non-terminal grammar elements like ⟨string⟩ and ⟨integer⟩ are not described.

Supply Side

To represent capabilities that apply to a specific buyer (perhaps for contractual reasons), we have defined a capability to be a list of ⟨buyerSpecificCapability⟩. With one exception, a buyer-specific capability applies only to one buyer – that buyer associated in the id field of the ⟨buyerSpecificCapability⟩. The exception

occurs if the id field is * or wildcard. This indicates that the capability applies to all buyers. Using buyer-specific capabilities, suppliers can represent specific capabilities to certain buyers and generic capabilities applying to all other buyers. By not including a wildcard ⟨buyerSpecificCapability⟩ and only listing ⟨buyerSpecificCapability⟩s applicable to specific buyers, sellers can also represent the fact that they will trade only with a subset of all buyers. In cases where both the wildcard ⟨buyerSpecificCapability⟩ and a ⟨buyerSpecificCapability⟩ applicable to a specific buyer apply, the most specific ⟨buyerSpecificCapability⟩ is selected.

A schematic of a ⟨sellerSpecificPreference⟩ is given in Figure 2.

We begin at the top level of a capability:

$$\text{capability} \rightarrow \{\langle\text{buyerSpecificCapability}\rangle\}$$

where

$$\begin{aligned}\langle\text{buyerSpecificCapability}\rangle \rightarrow\ & \mathsf{id}\text{: } \langle\text{id}\rangle,\\ & \mathsf{discrete}\text{: } \{\langle\text{discreteVarDescription}\rangle\},\\ & \mathsf{continuous}\text{: } \{\langle\text{continuousVarDescription}\rangle\},\\ & \mathsf{range}\text{: } \{\langle\text{rangeVarDescription}\rangle\},\\ & [\mathsf{discreteConstraint}\text{: } \langle\text{discreteConstraint}\rangle],\\ & \mathsf{instance}\text{: } \{\langle\text{discreteCapabilityInstance}\rangle\}\\ & [\mathsf{aggregationParticipation}\text{: } 0|1].\end{aligned}$$

⟨id⟩ identifies a buyer or group of buyers. Individual buyers are represented by some unique identifier (say an integer) and the group of all buyers is identified by the wildcard character '*'. So we have

$$\langle\text{id}\rangle \rightarrow \langle\text{integer}\rangle \mid * .$$

aggregationParticipation is a Boolean flag giving the supplier's willingness to participate in aggregate orders to the identified buyer.

Each of the variable constituent components is described by

$$\begin{aligned}\langle\text{discreteVarDescription}\rangle \rightarrow\ & \mathsf{name}\text{: } \langle\text{integer}\rangle,\\ & \mathsf{allowedValues}\text{: } \{\langle\text{integer}\rangle\}\\ \langle\text{continuousVarDescription}\rangle \rightarrow\ & \mathsf{name}\text{: } \langle\text{integer}\rangle,\\ & \mathsf{min}\text{: } \langle\text{double}\rangle,\\ & \mathsf{max}\text{: } \langle\text{double}\rangle\\ \langle\text{rangeVarDescription}\rangle \rightarrow\ & \mathsf{name}\text{: } \langle\text{integer}\rangle..\end{aligned}$$

In its simplest form, a $\langle$discreteConstraint$\rangle$ is a list of more primitive constraints

$$\langle\text{discreteConstraint}\rangle \rightarrow \{\langle\text{primitiveDiscreteConstraint}\rangle\}$$

where each primitive constraint is composed as follows:

$$\begin{aligned}\langle\text{primitiveDiscreteConstraint}\rangle \rightarrow\ & \mathsf{name}\text{: } \langle\text{string}\rangle\\ & \mathsf{variables}\text{: } \{\langle\text{discreteVarName}\rangle\},\\ & \mathsf{includes}\text{: } 0 \,|\, 1,\\ & \mathsf{values}\text{: } \langle\text{integerMatrix}\rangle\end{aligned}$$

$\langle$discreteVarName$\rangle$ is the name of the discrete variable involved in the constraint

$$\langle\text{discreteVarName}\rangle \rightarrow \langle\text{integer}\rangle.$$

The includes field is a bit. If the bit is 1, then the combinations listed in the values field are the allowed values the variables may take on. If the bit is 0, then the combinations listed in values are the excluded combinations, i.e., everything in the powerset of the variables is allowed except those combinations listed in values. The order of the variable names is significant, since they will be assumed to be in the same order in values. If there are a variables involved in the constraint, and c constraints, then $\langle$integerMatrix$\rangle$ is an $a \times c$ matrix of integers:

$$\begin{aligned}\langle\text{integerMatrix}\rangle &\rightarrow \langle\text{integerVector}\rangle, \cdots, \langle\text{integerVector}\rangle\\ \langle\text{integerVector}\rangle &\rightarrow \langle\text{integer}\rangle, \cdots, \langle\text{integer}\rangle\end{aligned}$$

The $\langle$discreteCapabilityInstance$\rangle$ component is described by

$$\begin{aligned}\langle\text{discreteCapabilityInstance}\rangle \rightarrow\ & \mathsf{mask}\text{: } \langle\text{discreteVarMask}\rangle,\\ & [\mathsf{rangeCapability}\text{: } \{\langle\text{rangeVarInstance}\rangle\}],\\ & \mathsf{continuousCapability}\text{: } \langle\text{continuousCapability}\rangle\\ & \mathsf{continuousConstraints}\text{: } \langle\text{continuousConstraints}\rangle\end{aligned}$$

A $\langle$rangeVarInstance$\rangle$ defines a range variable and has the form

$$\begin{aligned}\langle\text{rangeVarInstance}\rangle \rightarrow\ & \mathsf{name}\text{: } \langle\text{integer}\rangle,\\ & \mathsf{min}\text{: } \langle\text{double}\rangle,\\ & \mathsf{max}\text{: } \langle\text{double}\rangle.\end{aligned}$$

The $\langle$discreteVarMask$\rangle$ relates to the discussion of 6.2.2. As in Table 3 we have

$$\langle\text{discreteVarMask}\rangle \rightarrow \{\langle\text{extendedVarValue}\rangle\}$$

where an ⟨extendedVarValue⟩ is either an integer from the domain of the discrete variable or the wildcard character '*':

$$\langle \text{extendedVarValue} \rangle \rightarrow \ \langle \text{integer} \rangle \mid * \, .$$

⟨continuousConstraints⟩ describes the hard linear constraints for the continuous variables. Since these constraints may be either inequality or equality, we have

$$\begin{aligned} \langle \text{continuousConstraints} \rangle \rightarrow & [\mathsf{equality}\text{: } \langle \text{linearConstraints} \rangle], \\ & [\mathsf{inequality}\text{: } \langle \text{linearConstraints} \rangle] \end{aligned}$$

Both the equality and inequality constraints are expressed through a matrix which is $c \times n_c$ where c is the number of constraints, and a vector which is $c \times 1$. Consequently we have

$$\begin{aligned} \langle \text{linearConstraints} \rangle \rightarrow & \ \mathsf{matrix}\text{: } \langle \text{doubleMatrix} \rangle, \\ & \ \mathsf{vector}\text{: } \langle \text{doubleVector} \rangle \end{aligned}$$

A ⟨doubleMatrix⟩ is defined by

$$\langle \text{doubleMatrix} \rangle \rightarrow \langle \text{doubleVector} \rangle, \cdots, \langle \text{doubleVector} \rangle$$

and a ⟨doubleVector⟩ is just what the name suggests – a vector of doubles:

$$\langle \text{doubleVector} \rangle \rightarrow \ \langle \text{double} \rangle, \cdots, \langle \text{double} \rangle.$$

The only remaining undescribed element above is ⟨continuousCapability⟩ whose description is

$$\begin{aligned} \langle \text{continuousCapability} \rangle \rightarrow & \ \mathsf{breakPoints}\text{: } \langle \text{doubleListMatrix} \rangle, \\ & \ \mathsf{valAtBreakPoints}\text{: } \langle \text{doubleListMatrix} \rangle \end{aligned}$$

⟨doubleListMatrix⟩ describes a $n_c \times n_c$ matrix whose elements are lists of ⟨double⟩:

$$\begin{aligned} \langle \text{doubleListMatrix} \rangle & \rightarrow \langle \text{doubleListVector} \rangle, \cdots, \langle \text{doubleListVector} \rangle \\ \langle \text{doubleListVector} \rangle & \rightarrow \langle \text{doubleList} \rangle, \cdots, \langle \text{doubleList} \rangle \\ \langle \text{doubleList} \rangle & \rightarrow \{ \langle \text{double} \rangle \} \end{aligned}$$

It is assumed that the rows and columns of the matrix are in some canonical order so that we know which continuous variable is referenced. A natural order is the one defined in {⟨continuousVarDescription⟩}

Preferences

Just as capabilities may be buyer-specific so too may preferences be seller-specific. The same rules determining which seller-specific preference to apply are followed. A schematic of a ⟨sellerSpecificPreference⟩ is given in Figure 3.

We define a preference as follows

$$\langle\text{preference}\rangle \rightarrow \{\langle\text{sellerSpecificPreference}\rangle\}[, \langle\text{aggregatedPreference}\rangle]$$

i.e., a preference is a list of ⟨sellerSpecificPreference⟩ with an optional aggregated preference. We first describe ⟨sellerSpecificPreference⟩ and then consider ⟨aggregatedPreference⟩.

The ⟨sellerSpecificPreference⟩ is composed as follows

$$
\begin{aligned}
\langle\text{sellerSpecificPreference}\rangle \rightarrow\ & \mathsf{id}\text{: } \langle\text{id}\rangle, \\
& \mathsf{discrete}\text{: } \{\langle\text{discreteVarDescription}\rangle\}, \\
& \mathsf{continuous}\text{: } \{\langle\text{continuousVarDescription}\rangle\}, \\
& \mathsf{range}\text{: } \{\langle\text{rangeVarDescription}\rangle\}, \\
& \mathsf{dimensionWeights}\text{: } \langle\text{dimensionWeights}\rangle, \\
& \mathsf{discreteTradeoff}\text{: } \langle\text{tradeoffTables}\rangle \\
& [\mathsf{discreteConstraint}\text{: } \langle\text{discreteConstraint}\rangle], \\
& \mathsf{instance}\text{: } \{\langle\text{discretePreferenceInstance}\rangle\}
\end{aligned}
$$

Of these elements, only ⟨dimensionWeights⟩, ⟨tradeoffTables⟩, and ⟨discretePreferenceInstance⟩ have yet to be defined. ⟨dimensionWeights⟩ gives the weights of all dimensions that indicate their importance. For convenience we break up the weights according to the three types of variables. Thus we have

$$
\begin{aligned}
\langle\text{dimensionWeights}\rangle \rightarrow\ & \mathsf{range}\text{: } \langle\text{doubleVector}\rangle, \\
& \mathsf{discrete}\text{: } \langle\text{doubleVector}\rangle, \\
& \mathsf{continuous}\text{: } \langle\text{doubleVector}\rangle
\end{aligned}
$$

A ⟨doubleVector⟩ has been described previously. Each of the corresponding vectors is as long as the number of range, discrete, or continuous dimensions. ⟨tradeoffTables⟩ is an $n_{\text{discrete}} \times n_{\text{discrete}}$ matrix of ⟨tradeoffTable⟩:

$$
\begin{aligned}
\langle\text{tradeoffTables}\rangle &\rightarrow \langle\text{tradeoffTableMatrix}\rangle \\
\langle\text{tradeoffTableMatrix}\rangle &\rightarrow \langle\text{tradeoffTableVector}\rangle, \cdots, \langle\text{tradeoffTableVector}\rangle \\
\langle\text{tradeoffTableVector}\rangle &\rightarrow \langle\text{tradeoffTable}\rangle, \cdots, \langle\text{tradeoffTable}\rangle
\end{aligned}
$$

A ⟨tradeoffTable⟩ is simply a matrix of double values.

Finally, we turn to the last undefined component of a ⟨preference⟩. A ⟨discretePreferenceInstance⟩ is

composed as follows:

⟨discretePreferenceInstance⟩ → mask: ⟨mask⟩,
[rangeIdeal. {⟨rangeVarInstance⟩}],
continuousIdeal: ⟨doubleVector⟩,
tradeoffMatrix: ⟨doubleMatrix⟩,
[continuousConstraints: ⟨continuousConstraints⟩]

The rangeIdeal and continuousIdeal fields give the desired range and continuous trade parameters. The tradeoffMatrix field gives the positive definite matrix expressing the tradeoffs amongst the continuous variables. ⟨continuousConstraints⟩ have been described previously in the sell-side specification.

To complete the specification of preferences, we conclude with the definition of ⟨aggregatedPreference⟩. Refer to the discussion of section 6.1.7 for details.

⟨aggregatedPreference⟩ → participants: {⟨aggSpecification⟩},
contributionType: ⟨contributionTypeVector⟩,
additionalConstraints: ⟨continuousConstraints⟩,
discrete: {⟨discreteVarDescription⟩},
continuous: {⟨continuousVarDescription⟩},
range: {⟨rangeVarDescription⟩},
dimensionWeights: ⟨dimensionWeights⟩,
discreteTradeoff: ⟨tradeoffTables⟩
[discreteConstraint: ⟨discreteConstraint⟩],
instance: {⟨discretePreferenceInstance⟩}

In the above definition, the previously defined elements maintain their meaning. The additionalConstraints field allows the buyer to express constraints around the aggregation, such as "all orders must arrive on the same day," etc. participants lists the suppliers who can participate in the aggregation and their characteristics. Note that if the wildcard supplier participates, the order can potentially be aggregated across all suppliers. ⟨aggSpecification⟩ describes information specific to a supplier participating in the aggregation. It is defined by

⟨aggSpecification⟩ → id: ⟨id⟩.

id identifies the participating supplier and constraints specific to that supplier defined in an accompanying ⟨sellerSpecificPreference⟩ will be used in the optimization. Additional information may be added as required.

discrete mask	output	specificity
$\begin{bmatrix} * & * & * \end{bmatrix}$	{continuous1, range1}	0
$\begin{bmatrix} \varkappa_1 & * & * \end{bmatrix}$	{continuous2, range2}	1
$\begin{bmatrix} \varkappa_1 & * & \varkappa_3 \end{bmatrix}$	{continuous3, range3}	2

Table 3: Example of discrete masks for specifying continuous and range variables which are dependent on discrete variables. $\varkappa_1$ and $\varkappa_3$ signify specific values for the first and third discrete variables. The specificity of each mask is indicated in the third column.

The contributionType field is used to define the $\boldsymbol{\xi}$ vectors used in aggregation. The ⟨contributionTypeVector⟩ consists of n_c elements indicating the type of aggregation for each continuous dimension:

$$\langle\text{contributionTypeVector}\rangle \to \langle\text{contributionType}\rangle, \cdots, \langle\text{contributionType}\rangle.$$

Possible contribution types include

$$\langle\text{contributionType}\rangle \to \mathsf{sum},\ \mathsf{average},\ \mathsf{zero}.$$

sum sets $\boldsymbol{\xi} = \mathbf{1}$, average sets $\boldsymbol{\xi} = \mathbf{1}/|\mathcal{K}(\boldsymbol{\varkappa})|$, and zero sets $\boldsymbol{\xi} = \mathbf{0}$.

Masking

We have allowed constraints, ideal values, tradeoffs, and continuous capabilities to be dependent on discrete variables. In this section we describe an efficient manner in which to encode this dependence.

The data structure must represent continuous and range variables for all valid discrete inputs. An efficient way to do this is to use hierarchical definitions. At the top of the hierarchy are the definitions of the continuous and range variables for the discrete values $\boldsymbol{\varkappa}^t = [*, \cdots, *]$. These values apply to all $\mathbf{v}$ unless more specialized masks are defined. A more specialized mask of the continuous and range variables is specified by defining values for some of the components $\varkappa_\imath$. The more components that are defined, the more specialized the definition. The most specific mask is always used. An example definition for three discrete variables is given in Table 3. The response to the lookup $[\tilde{\varkappa}_1\ \tilde{\varkappa}_2\ \tilde{\varkappa}_3]$ is {continuous3, range3} if and only if $\varkappa_1 = \tilde{\varkappa}_1 \wedge \varkappa_3 = \tilde{\varkappa}_3$, {continuous2, range2} if and only if $\varkappa_1 = \tilde{\varkappa}_1 \wedge \varkappa_3 \neq \tilde{\varkappa}_3$, and {continuous1, range1} otherwise.

Match Results

Returned to the buyer is a list of matches with different suppliers, which can be ranked and viewed in many different ways in the GUI. A ⟨matchResultList⟩ is a list of matchResult:

$$\langle\text{matchResultList}\rangle \to \{\langle\text{matchResult}\rangle\}.$$

A match result may either be a ⟨singleSupplierMatchResult⟩ or an ⟨aggregatedMatchResult⟩:

$$\langle\text{matchResult}\rangle \rightarrow \langle\text{singleSupplierMatchResult}\rangle \,|\, \langle\text{aggregatedMatchResult}\rangle.$$

A ⟨singleSupplierMatchResult⟩ represents the best trade with a single supplier and is composed of the following elements:

$$\begin{aligned}\langle\text{singleSupplierMatchResult}\rangle \rightarrow\ & \mathsf{supplierId}\text{: } \langle\text{integer}\rangle,\\ & \mathsf{utility}\text{: } \langle\text{double}\rangle,\\ & \mathsf{feasible}\text{: } 0|1,\\ & [\mathsf{costFactors}\text{: } \{\langle\text{double}\rangle],\\ & \mathsf{continuous}\text{: } \{\langle\text{double}\rangle\},\\ & \mathsf{discrete}\text{: } \{\langle\text{discreteVarDescription}\rangle\},\\ & \mathsf{range}\text{: } \langle\text{rangeVarInstance}\rangle.\end{aligned}$$

The supplierId indicates the supplier sourcing this trade and the utility field indicates the utility of the trade (which can be used to rank the trades). feasible is a bit indicating whether or not a feasible trade with this supplier was found. The continuous, discrete, and range fields list the respective trade parameters determined by the matching algorithm. The optional cost factors field lists the constituent costs contributing to the total cost of ownership C_o evaluated at the trade point returned in the ⟨singleSupplierMatchResult⟩.

An ⟨aggregatedMatchResult⟩ returns the optimal trade when the buyer has requested aggregation. It is composed of the following elements:

$$\begin{aligned}\langle\text{aggregatedMatchResult}\rangle \rightarrow\ & \mathsf{utility}\text{: } \langle\text{double}\rangle,\\ & \mathsf{feasible}\text{: } 0|1,\\ & [\mathsf{costFactors}\text{: } \{\langle\text{double}\rangle],\\ & \mathsf{supplierTradeParameters}\text{: } \{\langle\text{supplierTradeParameters}\rangle\}.\end{aligned}$$

As before, the utility field gives the utility of the aggregate trade, and the feasibility flag indicates whether or not a feasible aggregate trade was found (there may not be if the constraints were too stringent). costFactors can also be returned in C_o is sufficiently general to handle aggregated trades. Finally, ⟨supplierTradeParameters⟩ lists the trade parameters for each supplier involved in the aggregation. It is defined as follows:

$$\begin{aligned}\langle\text{supplierTradeParameters}\rangle \rightarrow\ & \mathsf{supplierId}\ \langle\mathit{integer}\rangle,\\ & \mathsf{continuous}\text{: } \{\langle\text{double}\rangle\},\\ & \mathsf{discrete}\text{: } \{\langle\text{discreteVarDescription}\rangle\},\\ & \mathsf{range}\text{: } \langle\text{rangeVarInstance}\rangle.\end{aligned}$$

6.3 Summary

We have described an efficient computational procedure in which to encode buyer's trading preferences and hard constraints, supplier's delivery capabilities and constraints, and optimize to find those matches between one buyer and one or many sellers that maximize the buyer's utility. By optimizing against both qualitative and quantitative factors, and exploiting the trading flexibilities possessed by both parties, the system determines better trades. The tool is particularly useful as companies move their direct material purchasing online. By optimizing across flexibilities, win-win trades are discovered for both trading parties.

The representation of trading preferences is designed to be expressive yet easily elicitable from a buyer, and computationally tractable. The representation of supplier capabilities was chosen to parallel the manner in which suppliers already think of their delivery capabilities and seamlessly includes volume discounts and incremental costs. These supplier capabilities may be part of an online catalog. The representation of the negotiation space is rich, supporting three types of variables.

We have outlined a manner in which preferences may be inferred automatically through models of the purchasing company. Such models incorporate many cost factors, taking the total cost of ownership into account. The system provides trades which minimize the total cost and represent significant new savings.

The invention can operate both at a buyer's site, where suppliers input their capabilities through an HTML interface to the world wide web or as an embedded part of an electronic market hosted by a particular web site. The invention may operate at regularly scheduled intervals or sporadically in lieu of current request for quotations (RFQ). The buyer may broadcast a RFQ event to suppliers, indicating a time within which suppliers must respond. At the close of the event, the buyer can use the present invention to assist in the analysis of the supplier responses.

Complex algorithms have been specified which should permit most matching optimization to occur in near real-time. The rapidity of optimization, combined with graphical what-if tools, allows for analysis and exploration of trades, which should significantly improve the quality of purchasing decisions.

While the above invention has been described with reference to certain preferred embodiments, the scope of the present invention is not limited to these embodiments. One skilled in the art may find variations of these preferred embodiments which, nevertheless, fall within the spirit of the present invention, whose scope is defined by the claims set forth below.